OBLIVIOUS

BFS

&

DFS

OKVS : vertex → d-edges

↳ We have this
or basically we have graphs

We know O(V+E) ⇒ Normal Complexity

So, we should try for O(N+E) log E)
type of code.

But this would ideally mean that log(E)
is coming from hiding about the info?
Or can we say than ORAM Access.

1st IDEA

One Approach :

OKVS initialised.

Path ORAM ⇒ nodes are stored.
to retrieve the node. O(log E) accss
then, what we can work/on is O(E log E)
access.

What to Do? ← { 75% Dummy Op.
25% Real Op.

Qcnt = 1, curQcnt = 1, S ← Source

DomAP[visited] ← true

DomAP[Qnn] ← S

Qcnt ← Qcnt + 1

p ← Qkvs.read(S)

Q.append(p)

while (curQ ≠ Qcnt) do

// D nodes appended together

[Saves us from Expensive DomAP access]

Every Read will give d neighbours, reducing lookups O(E) to O(E/d)

Enqueuing d at once go instead of 1.

Dummy Implement.

Around 1/4th of edges are real & rest are dummy.

After 'n' operations n·d queue elements.

X = Total no. of real entries

p = 1/4, dummy = 3/4

$2V \times \left(\left\lceil \frac{2E}{V} \right\rceil + 1 \right)$

$P(\text{Edge is Real}) = 1/4$

Binomial $(n \cdot d, p)$

$\mu = \frac{n \cdot d}{4}$

$$Pr[X > (1+\delta)\mu] < e^{-\frac{\delta^2 \mu}{3}} = e^{-\frac{\delta^2 nd}{12}}$$

We fix enough of δ:

i.e.

V	E	d	ℓ
⋮	-	⋮	⋮

So, now we will get fixed δ.

↳ Get the minimal value with high Pr. value.

↳ Guess this would improve the algorithm.

IDEA

Visited Array

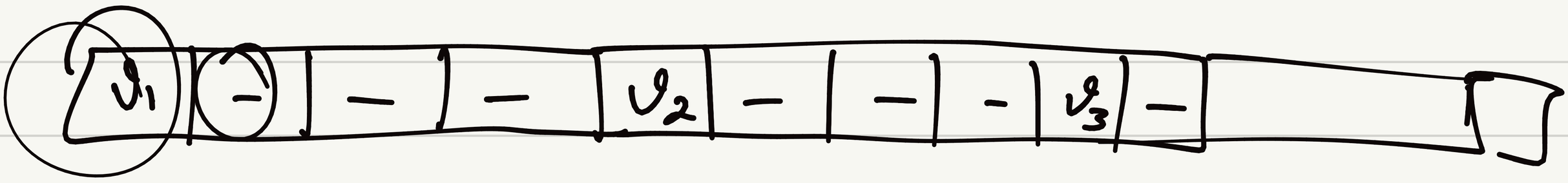
[1] , [0] ← Present in Queue or not

↳ visited or not

So, Algo.

$O(2V) \cdot [O(\log 2V) \cdot O(d) + O(PQ)]$

↑ Big Loop ↑ ORAM Visit Check ↑ After n iterations



Initialize: OKVS: 2V x d sized structure
Visited Array (V) : 2 bits
V[i] = 00 ← Placed in queue or not
↑ Processed or not

Queue of size "Consider 2".

Source Vertex = 's'

So,
dqt = 1, CurDqt = 1
source ← start

DOMAP[("InQ", dqt)] ← start
ORAM[("Visited", source)] ← 11

dqt ← dqt + 1
[[xq]] ← [[source]]

for num in range 1 to 2V do:
[[ei]] ← KVS.read [[xq]]
Some_func. ([[ei]])
if [[continue?]] then
Pretend to Access the Visited Array
$x_q \leftarrow x_{q+1}$
else:
Visited [v] = 11
end if
Queue handling
Confection:

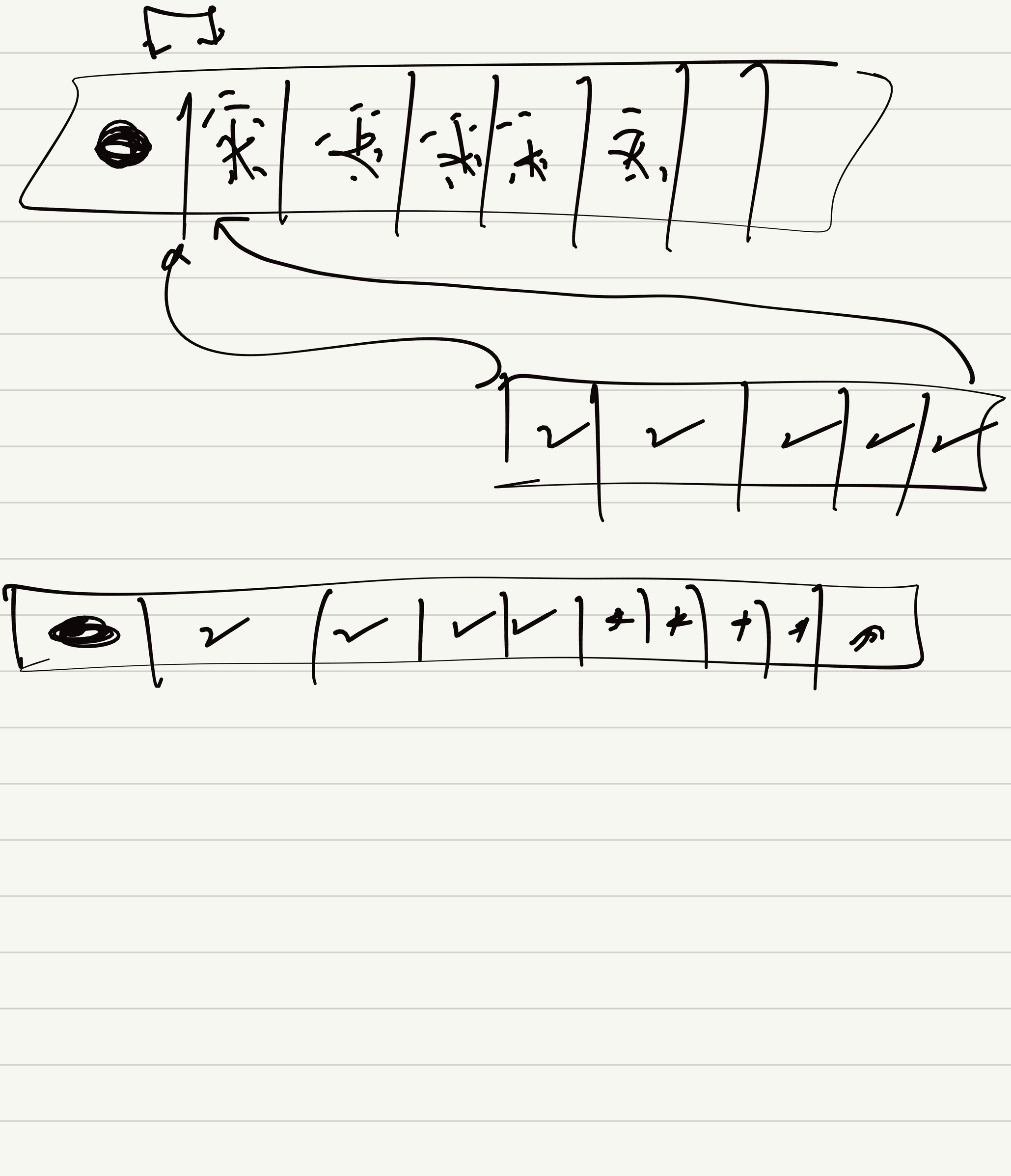
Eg:

Consider: $d = 2$ $n = 100$

$l = 75$ $x > 75 \Rightarrow$ Removed

while $1 + 200 - 100 = 101$ elements

1/4 Reduced



* We only place those vertex in our DOMAP Queue which has not yet been accessed, but for that we can do something like:

v_1	v_2	v_3	v_4	----	v_{deg-2}	v_{deg-1}	v_d
0	0	0	1		1	0	0

Sort them with Insertion

$O(d) \times O(\text{Sort}) \times O(\text{Place it in Queue})$

$O(\log^2 V) \times O(d) \times O(\text{Write in Some Other Array})$

$+ \quad O(\text{This Array}) \times O(\text{d index}) \times O(\text{curr. index})$

One more idea is that: we can keep

while loop with condition:

dcnt ≠ curdcnt or count == E

But we can show that worst case

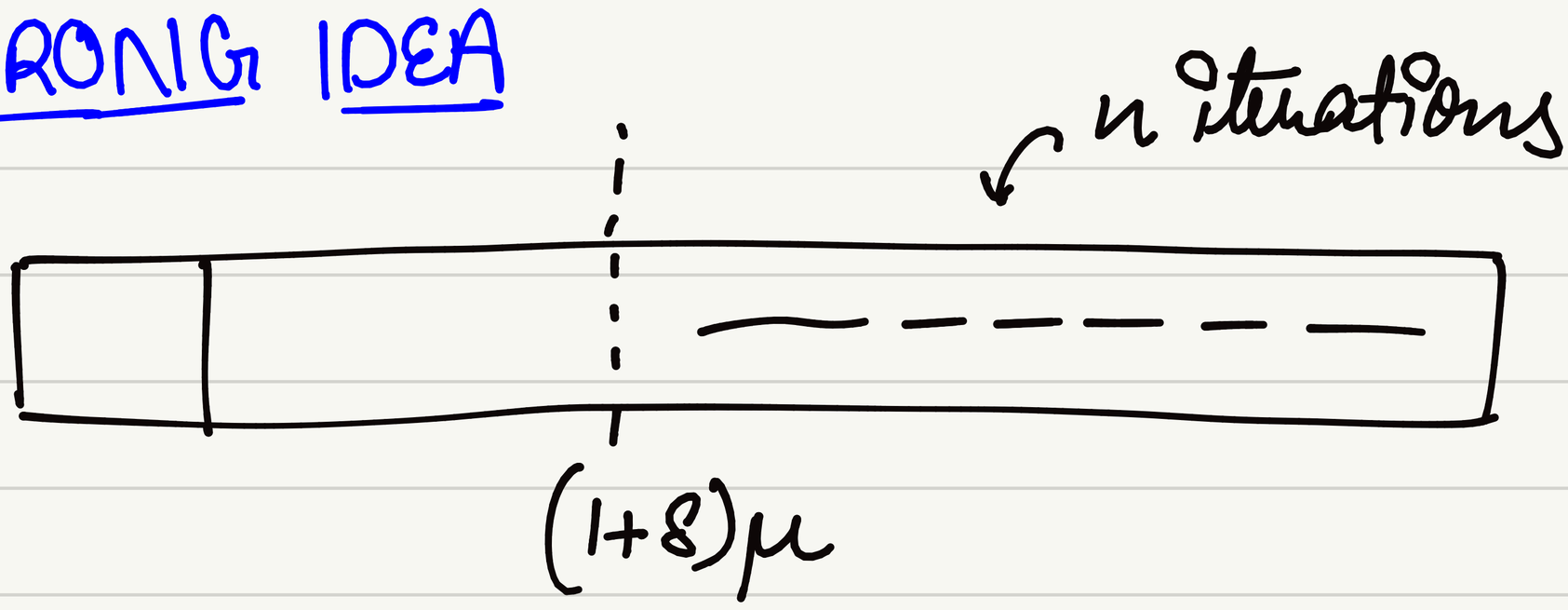
Not Feasible { Cord. would be $E \times d \times \log^2 V$

& avg. or best case $V \times d \times \log^2 V$ }

Would this suffice?

I'd have to think about that.

WRONG IDEA



(V, E) : $P(\text{Real}) = 1/4$, $P(\text{Dummy}) = 3/4$

Q_{k-1} : Queue size immediately after $(k-1)^{th}$ compaction

Then, n iterations before k^{th} compaction

Queue size before Compaction

$$L_k = Q_{k-1} + n(d-1)$$

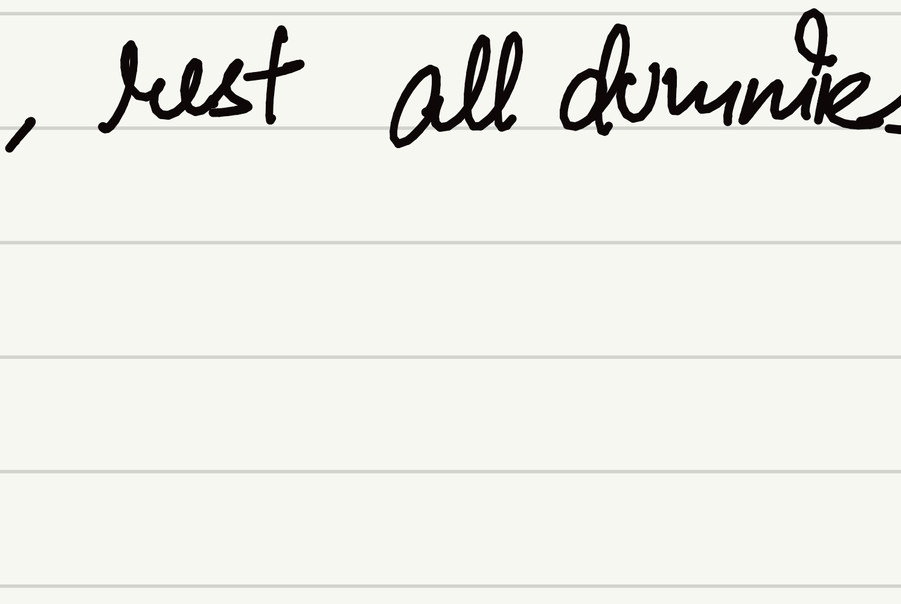
Front 't_k' real edges, rest all dummies

$$t_k = (1+\varepsilon)\frac{L_k}{4}$$

$X = \sum X_i$ i.i.d

ε

Accessed Only Once



(v) $v \times \frac{v}{2}$ $\frac{v^2}{2}$ → Size increase

This is not fair.

I don't think this would work.

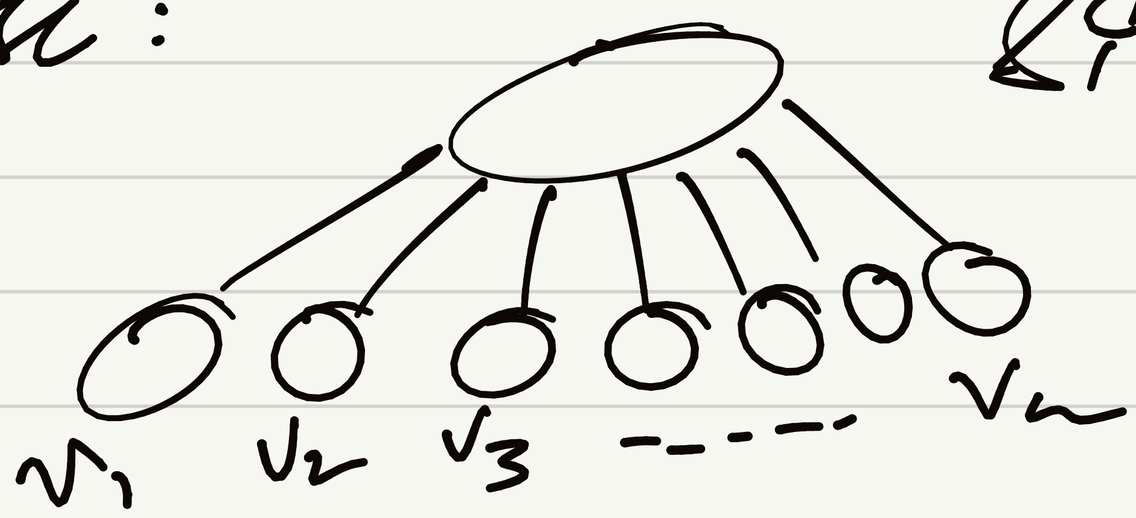
Compaction:

I think the chernoff bounds would work perfectly with the given values but for safety what we can come up is,

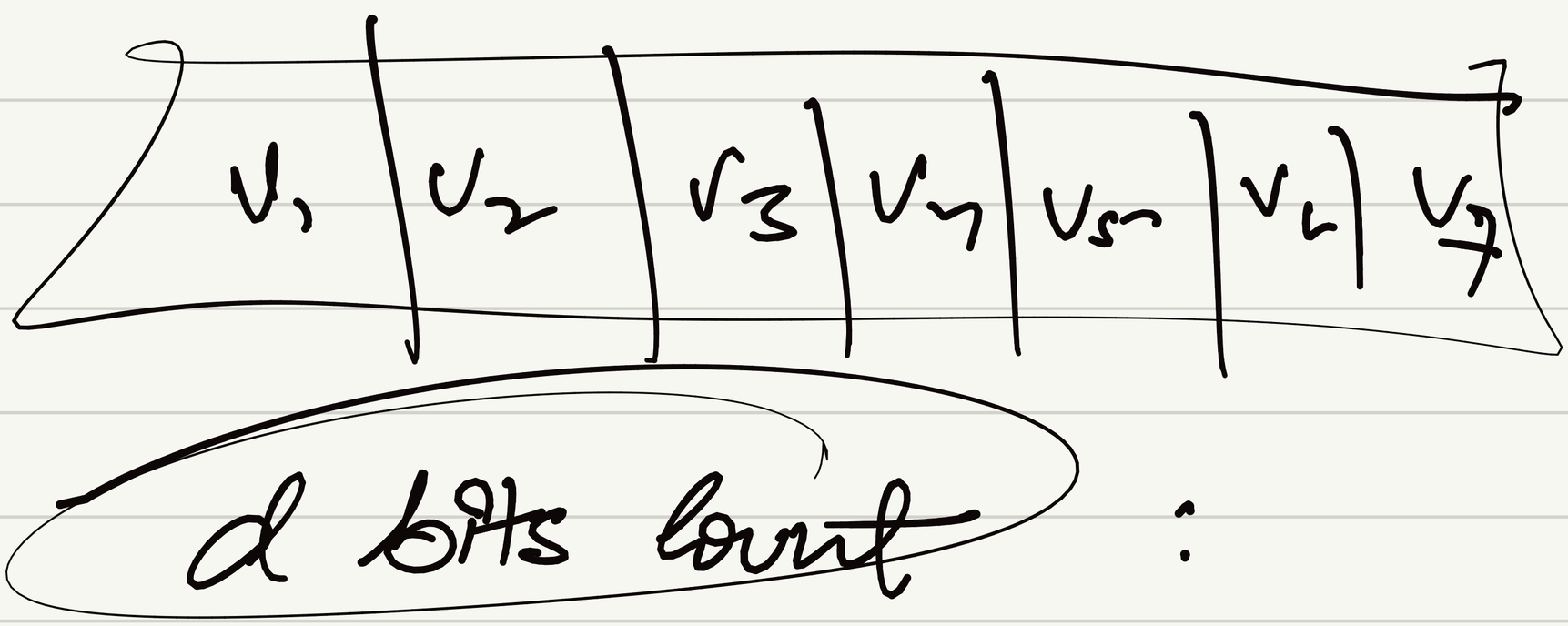
till 'n' iterations or if queue size > 2E. → Compact it to E.

Consider the case

Case:



Single Node, connected to all the other vertices.



d bits count :

↳ keep a count of nos. in the queue as well as keep a count of

Graph Size = $4G$

Real Graph Size = G

$p = \frac{1}{4}$ [Prob. that a chosen edge is real]

Let after n iterations we are doing compaction.

So, Distribution ~ $B(n \cdot d, p)$

$$\mu = \frac{nd}{4}$$

We know,

$$Pr[X > (1+\delta)\mu] < e^{-\delta^2\mu/3}$$

r.v. X = Total no. of Real Entries

$$Pr[X > (1+\delta)\mu] < e^{-\delta^2\mu/3} \leq 2^{-80}$$

So, $e^{-\delta^2\mu/3} \leq 2^{-80}$

$$\Rightarrow \frac{\delta^2\mu}{3} \geq 80 \ln 2$$

$$\Rightarrow \frac{\delta^2 \cdot nd}{12} \geq 80 \cdot \ln 2$$

$$\Rightarrow \quad \delta^2 \geq \frac{80 \cdot 12 \cdot \ln 2}{nd}$$

$$\therefore \quad \boxed{\delta \geq \sqrt{\frac{960 \cdot \ln 2}{nd}}}$$

$$\delta_0, \quad \delta_{min} = \sqrt{\frac{960 \cdot \ln 2}{nd}}$$

Now, we have to check $(1+\delta)\mu$

$$\delta_0, \quad (1+\delta)\mu \geq \left(1 + \sqrt{\frac{960 \times \ln 2}{nd}}\right) \frac{nd}{4}$$

$$= \frac{nd}{4} + \cancel{4} \times \sqrt{60 \times \ln 2} \times \frac{nd}{\cancel{4}}$$

$$\therefore \left\{ (1+\delta)\mu \geq \underbrace{\frac{nd}{4} + \sqrt{60 \times \ln 2 \times nd}} \right\}$$

So, our cost func^n is dependent upon n, since for a given graph, value of 'd' is fixed.

OBLIVIOUS BFS & DFS: GRAPH OS:

$O(C \log^2|E| \cdot \log^2 \log |E|) = O(\alpha)$

1: function Execute(DOMAP, start)
◁ BFS-specific parts are in red and DFS-specific ones in blue
2: Qcnt = 1; curQCnt = 1;source ← *start* O(1)
3: DOMAP [("*InQ*", Qcnt)] ← *start* $O(C \log^2|E| \log^2(\log|E|))$
4: DOMAP [("*Visited*",source)] ← *true* $O(C \log^2|E| \cdot \log^2(\log|E|))$
5: Qcnt ← Qcnt + 1 ; Qcnt ← Qcnt O(1)
6: outerL = *true* O(1)
7: while curQCnt ≠ Qcnt; Qcnt ≠ 0 do [Each elem. is deque & every edge is connected to, E itr.
8: tmp ← DOMAP [("*InQ*", curQCnt; Qcnt)] O(α)
9: source ← *Osel*(outerL, tmp,source) O(1)
10: curQCnt ← *Osel*(outerL, curQCnt + 1, curQCnt) O(1)
11: Qcnt ← *Osel*(outerL, Qcnt − 1, Qcnt) O(1)
12: cnt ← *Osel*(outerL, 1, cnt) O(1)
13: trm ← DOMAP [("*EOut*",source, cnt)] O(α)
14: outerL ← *Osel*(trm == *NU LL*, *true*, *f alse*) O(1)
15: tmp ← DOMAP [("*Visited*", trm)] O(α)
16: visited ← *Osel*(outerL, visited, tmp) O(1)
17: mostInner ← visited = *NU LL* & outerL = *f alse* O(1)
18: tmp ← *Osel*(mostInner, trm, *dummy*) O(1)
19: DOMAP [("*InQ*", Qcnt; Qcnt + 1)] ← tmp O(α)
20: tmp ← *Osel*(mostInner, *true*, *dummy*) O(1)
21: DOMAP [("*Visited*", trm)] ← tmp O(α)
22: Qcnt ← *Osel*(mostInner, Qcnt + 1, Qcnt) O(1)
23: cnt ← *Osel*(outerL, cnt, cnt + 1) O(1)
24: end while
25: end function

8:	Fetch Current Queue Node	1 FIND
13:	Fetch Outgoing Edge	1 FIND
15:	Check Visited	1 FIND
19:	Insert into Queue (if needed)	1 INSERT
21:	Mark visited	1 INSERT

Complexity:

- Find Accesses over all edges:

$$O(E) \times O(C \log^2 |E| \cdot \log^* \log(E)) = O(C\,|E| \log^2 |E| \log^* \log E)$$

- Insert Accesses (visited + queue):

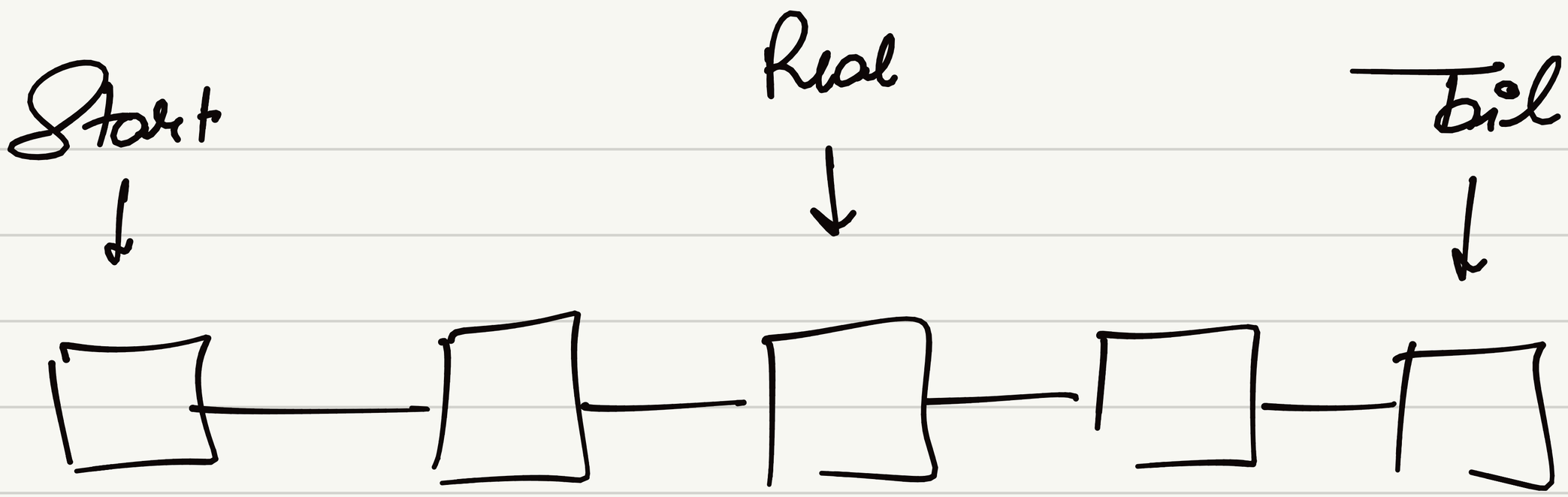
$$O(V) \times O(\alpha) = O(C\,|V| \log^*(E) \log^*(\log E))$$

BFS Complexity:
Each iter. does 2 FIND + 1 INSERT:

$$O\left(C\,(|V|+|E|) \log^2 |E| \cdot \log^* \log |E|\right)$$

{ Total iterations = $O(|E|)$
Total insertions = $O(|V|)$ }

Per iteration = 1 edge processed → 2-3 DOMAP Accesses



Can we implement the Queue as Linked List?

NO, as this could leak imp. info on Access Pattern.

Compaction

$S_k \subseteq [1, V]$: Set of already seen vertices upto compaction k

$|S_k| = s_k$ {No. of vertex processed}

$u_k = V - s_k$

Each edge is uniformly sampled from the vertex set

$$\left\{ \text{So}, \quad q_k = \frac{u_k}{V} = 1 - \frac{s_k}{V} \right\} \quad \text{—(1)}$$

Compaction 'k';

t_k = Real Entries

$$E[\text{new vertices}] = ud \cdot p \cdot q_k$$

So,

$$s_{k+1} = s_k + \underbrace{ud p}_{a} \left(1 - \frac{s_k}{V}\right)$$

$$\Rightarrow s_{k+1} = s_k \left(1 - \frac{a}{V}\right) + a$$

Linear Recursive Eqn:

$$S_k = v\left(1-\left(1-\frac{a}{v}\right)^k\right)$$

R_k = Exp. no. of new real entries added in k^{th} Connection.

$$R_k = nd\cdot p\cdot\left(1-\frac{S_k}{v}\right) = a\left(1-\frac{S_k}{v}\right)$$

$$= a\left(1-\left[1-\left(1-\frac{a}{v}\right)^k\right]\right)$$

$$\boxed{R_k = a\cdot\left(1-\frac{a}{v}\right)^k}$$

$$t_{k+1} = t_k + R_k$$

$v = a$

$ndp = v$

$$\boxed{n = \frac{v}{dp}}$$

$t_k \leq t_{max}$

$R_k \approx 0$ for that $\Rightarrow S_k \sim v$

$S_k \geq (1-\varepsilon)v$ $\quad \varepsilon \to 2^{-8}$

$$\left(1-\frac{a}{v}\right)^k \leq \varepsilon$$

$$\Rightarrow k \geq \frac{\ln\varepsilon}{\ln\left(1-\frac{a}{v}\right)} \approx \frac{v\left(\ln(\varepsilon)\right)}{a}$$

• a_k : Real Entries after Compaction k

• s_k : Dist. vertices seen

• New Real Entries X_k — Bin(--)

$$t_k = a_{k-1} + n(d-1)$$

$$a_k = \min(t_k, t_{max})$$

$$v(1-\alpha^k)$$

$$Pr[X_k > t_{max} - a_{k-1}] \leq \varepsilon$$

$$s_k = v\left(1-\left(1-\frac{a}{v}\right)^k\right)$$

increasing

$$E[X_k] = ndpq_k$$

$$\underbrace{\mu_k = ndpq_k}_{\text{decreasing}}, \quad \delta_k = \frac{t_{max} - a_{k-1}}{\mu_k} - 1$$

$$Pr[X_k > (1+\delta_k)\mu_k] \leq e^{-\frac{\delta_k^2}{3}\cdot\mu_k} \leq c^{-\omega}$$

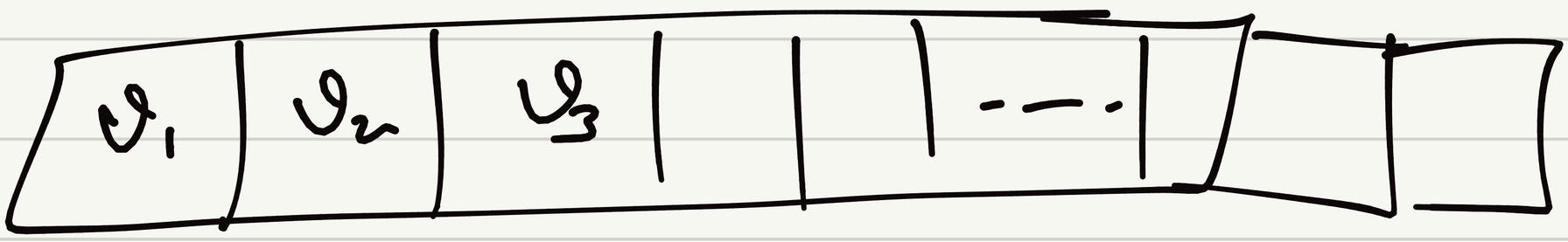
$$\mu_k = ndpq_k \leq ndpq_1 = ndp$$

$$\delta_k = \frac{t_{max} - a_{k-1}}{\mu_k} - 1 \geq \frac{t_{max}}{ndp} - 1 = \delta_1$$

It seems like if chernoff bounds holds for first compaction, then it holds for all later compaction.

This idea wouldn't work since this would make the queue size to be O(V) but we need an algo. which has a queue size of sublinear of V.

Worked On Another Idea but this doesn't work



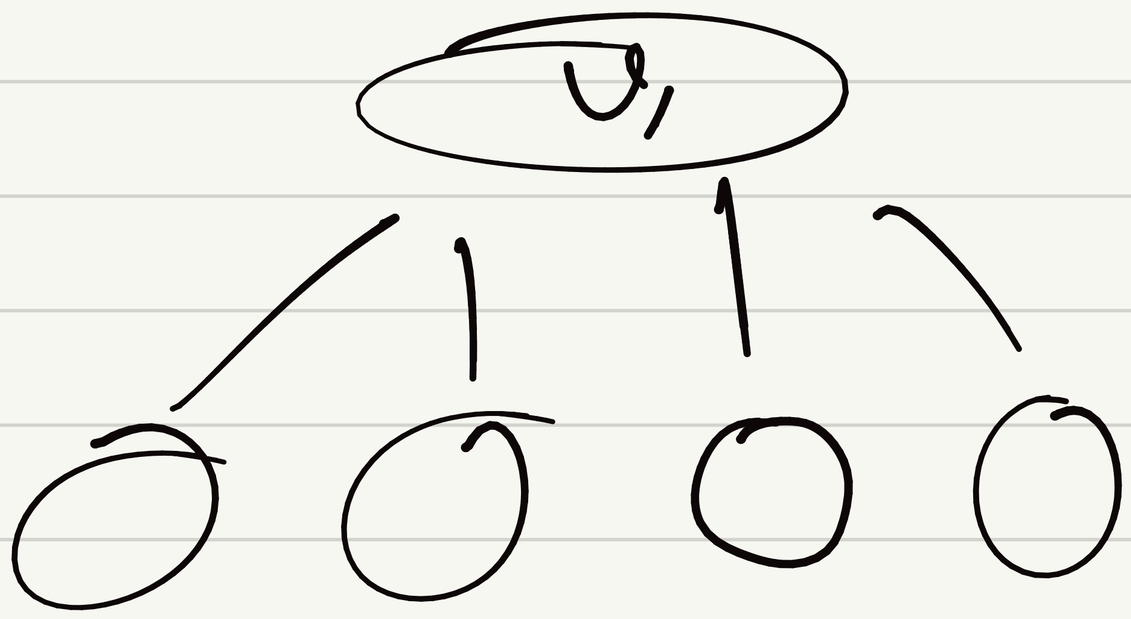
S_K : Real Entries after 'k' Compactions

S_{K+1} : Real Entries after 'k+1' Compactions

X : Entries that come in between

$\{S_K + Pad = d\}$

Queue Size $\leq v$



$\rightarrow d + n(d-1) = t_{K+1}$

Size of Queue before '$K+1^{th}$ Compaction

$s_k + a_k = d$

a_k: Padding of Queue till desired size to

$d + n(d-1) = t_{k+1}$

$t_{k+1} \Rightarrow d$

This Compaction isn't Working for Worst Case Scenario.

$s_k \rightarrow$ Can we say anything about this with dummy operation in play.

$a_k = d - a_k$

a_k: Real Entries Currently Present in the Queue

$O(V) + O(nd)$

$O(d)$

ndp_{S_K} → New Real Entries

q_K { U_K: Real Entries Present in Queue
S_K: Real Entries Already Processed

$\frac{q_K}{v} = \frac{U_K + S_K}{v}$: Pr [Getting Non Processed Ref]

X : Real Entries Going in Current Round

$$E[X] = ndp_{q_K} = ndp\left[1 - \left(\frac{U_K + S_K}{v}\right)\right]$$

$$= ndp\left[1 - \frac{q_K}{v}\right]$$

$$q_K = v\left(1 - \left(1 - \frac{a}{v}\right)^K\right)$$

* This can be used to get the Real Entries that are being appended in the list.

So,

$$Pr[X > t_{max} - a_{k-1}] \leq \varepsilon$$

$$\varepsilon \approx 2^{-80}$$

$$E[X_k] = q_k = ndp\, q_k = ndp\left[1 - \frac{a_k}{v}\right]$$

$$= ndp\left[\left(1 - \frac{a}{v}\right)^k\right]$$

$$d = \left\lceil \frac{2E}{V} \right\rceil + 1$$

$$(d-1) = \left\lceil \frac{2E}{V} \right\rceil$$

2V Rows ⇒ atleast 2V edges

So 2V edges

$d = 5$

d

$$5 = \left\lceil \frac{2E}{V} \right\rceil + 1$$

$$\left\lceil \frac{2E}{V} \right\rceil = 4$$

$$\left\lceil \frac{5V}{V} - 1 \right\rceil$$

atmost $V/2$ > d edges

atleast $V/2$ < d edges

$$\left\lceil d - 1 \right\rceil$$

$$d - \frac{2}{V}$$

$$\frac{V(d-1)}{2} = E$$ Min

for exactly v vertices, d edges list

$$\frac{V(d-1)}{2} \geq E$$

v vertices $\frac{d}{2}$ edges

ROUGH CALCULATIONS :

(x) → 1 edge

Max. edge for (V, d)

$$d = \left\lceil \frac{2E}{V} \right\rceil + 1$$

$$\boxed{\frac{(d-1)V}{2} = E}$$

So, we know $\boxed{x \geq V/2}$ $\boxed{d(x) < d}$

$$\frac{V(d-1)}{2} - x$$

Real Edges = $\frac{(d-1)V}{2}$ = (Vx)

$$\therefore \boxed{x = \frac{d-1}{2}}$$

So, $\boxed{d(x) = \frac{d-1}{2}}$

Suppose, $d=3$,

$$s(x) = 1$$

a_1 : Real vertices with $deg(v) \leq d$
a_2 : Real vertices with $deg(v) > d$ but $deg(v) \leq 2d$
⋮ ⋮ ⋮ ⋮

We know,

$a_i \geq 0$
$a_1 \geq V/2$
$a_i \leq V-1$

$a_1 \cdot 1 + a_2 \cdot 2 + a_3 \cdot 3 + \cdots = 2V$

$\sum_{i=1}^{\infty} a_i = V$ ↳ Vertices

↳ Edges

$a_1 \cdot 1 + a_2 \cdot (d+1) + a_3 \cdot (2d+1) + \cdots \leq \frac{V(d-1)}{2}$

Since, $d = \left\lfloor \frac{2E}{V} \right\rfloor + 1$

So, E_{max} for fix $V, d = \frac{V(d-1)}{2}$

I feel like its wrong

$$Pr[d_t > O(n)] \leq \varepsilon$$

$$\mu = E[X] = ndp$$

$$X = \sum_{i=1}^{n} X_i \; ; \; X_i \in [0, a]$$

$$Pr[T > (1+\delta)\mu] \leq e^{-\frac{\delta^2 \mu}{\delta+2}}$$

Suppose,

$$(1+\delta)\mu = cm$$

So, if $n = \Theta(m/dp)$

$$ndp(1+\delta) = cm \qquad \delta\mu = a - \mu$$

$$\delta = \frac{cm}{\mu} - 1$$

$$\frac{\delta\mu^2}{\delta+2} = \frac{(cm-\mu)\left(\frac{cm}{\mu} - 1\right)}{\left(\frac{cm}{\mu}+1\right)}$$

$$= \frac{(cm-\mu)(cm-\mu)}{(cm+\mu)}$$

μ μ

$$= -\frac{(cm-\mu)^2}{(cm+\mu)}$$

$$e^{-\frac{(cm-\mu)^2}{(cm+\mu)}} \leq 2^{-80}$$

$$\frac{(cm-\mu)^2}{(cm+\mu)} > 80(\ln 2)\ \alpha$$

$$\boxed{\mu = \frac{nd}{4}}$$

$$\mu^2 - 2cm\mu + c^2m^2 > \alpha cm + \mu\alpha$$

$$\Rightarrow \mu^2 - (2cm+\alpha)\mu + (c^2m^2 - \alpha cm) > 0$$

So, $\underline{\underline{D<0}}$ $\quad (2cm+\alpha)^2 - 4(c^2m^2 - \alpha cm) < 0$

$$\cancel{4c^2m^2} + 4cm\alpha + \alpha^2 - \cancel{4c^2m^2} + 4\alpha cm < 0$$

$$8cm\alpha + \alpha^2 < 0$$

$$\alpha(8cm+\alpha) < 0$$

(+ve, −ve)

$$\boxed{c < -\frac{\alpha}{8m}}$$

Never Possible

$$R_k = ndp\left(1-\frac{a}{v}\right)^k$$

Real Entries that are added after the k^{th} Dupaction.

So,

$$Q_k = R_k - n = ndp\left(1-\frac{ndp}{v}\right)^k - n$$

$$S = \sum_{k=0}^{\alpha} \widehat{ndp}\left(1-\frac{a}{v}\right)^k - n$$

$$S = a\sum_{k=0}^{\alpha}\left(1-\frac{a}{v}\right)^k - \alpha n$$

$$= a\left[\frac{1-(1-a/v)^{\alpha-1}}{a/v}\right] - \alpha n$$

$$\boxed{|S| = v\left[1-\left(1-\frac{a}{v}\right)^{\alpha-1}\right] - \alpha n}$$

α var., n var.

$$X = \sum_{i=1}^{N} X_i \quad ; \quad X_i \in \{0,1\} \qquad \mu$$

$$Pr[X_k \geq (1+\delta)\mu] \leq e^{\frac{-\delta^2 \mu}{\delta+2}}$$

$$E[X_k] = ndp\left[1 - \frac{a}{v}\right]^k \qquad \underbrace{a = ndp}$$

Then,

$$Pr[X_k \geq (1+\delta)\mu_k] \leq e^{\frac{-\delta^2 \mu_k}{\delta+2}}$$

Queue Size after compaction k:

$$Q_k = X_k - n$$

So, $Q_k \leq (1+\delta)\mu_k - n \leq \underbrace{c \cdot n}_{\text{Consider}}$

$$(1+\delta)ndp\left(1 - \frac{ndp}{v}\right)^k - n \leq c \cdot n$$

TRYING TO SHOW BOUND ON QUEUE SIZE

$$f(n) = n\left[\underbrace{dp(1+\delta)}_{\alpha}(1-x)^k - 1\right] \leq c \cdot n$$

$$f(n) = \alpha n\left(1 - an\right)^k - n \qquad \alpha = dp(1+\delta)$$

$$a = \frac{dp}{v}$$

$$f'(n) = \alpha(1-an)^k - k\alpha na(1-an)^{k-1} - 1$$

$$f''(n) = -k\alpha a(1-an)^{k-1} - k\alpha a(1-an)^{k-1} + k\alpha na^2(k-1)(1-an)^{k-2}$$

$$= -2k\alpha a(1-an)^{k-1} + k\alpha na^2(k-1)(1-an)^{k-2}$$

$$= k\alpha a(1-an)^{k-2}\left[-2(1-an) + an(k-1)\right]$$

$$\left[-2 + 2an + kan - an\right]$$

$$= \underbrace{\left[-2 + (k+1)an\right] k\alpha a (1-an)^{k-2}}$$

So, $\underbrace{\text{minima exists}}_{\text{+ve}}$ at a

$$f(n) = n\alpha e^{-kan} - n$$

$$f'(n) = -1 + \alpha e^{-kan} - n\alpha ka e^{-kan}$$

$$= -1 + \alpha(1 - nka)e^{-kan} = 0$$

$$\alpha(1-nka)e^{kan} = 1$$

MAIN ALGO.

FINAL ALGORITHM [BFS]

- Outer loop that iterates through every vertices ⇒ O(V)

- Every ORAM Access takes O($\log^2$V)

- Linear Scan ⇒ O(d)

- Compaction ⇒ O(d log d)

So, the idea is as follows:

- We will make an outer loop that would be running in O(V) time.

- Now, we would be maintaining an array in ORAM of size of O(V) that would be containing 2 bits, corresponding to every vertices.

- first bit shows whether it has been pushed into the queue or not while second is whether it has been processed in the queue or not.

- Now, whenever we place any new vertex into our queue, we would mark the first bit to 1 while second bit becomes '1' after it gets processed.

- So, for every loop iteration, we would first get the first element of our queue, pop it [O(1)] and then do an ORAM Access to mark this vertex visited.

- Now, for this vertex, we would be doing an OKVS Access, to get its neighbour.

- Then, we would be linear scanning all these vertices, to check whether any other vertex is already present in the queue or not. If present we would mark it as NULL or dummy. & then do a compaction.
- Now, we would place this 'd' list at the end of the queue.

- And our current idea is to do compaction after every 'k' rounds.

Complexity:

$$O(V) \cdot \Big[O(\log^2 V) + \big(O(d) \times O(\log^2 V) + O(d \log d) \big) + O(\log^2 V) + O\Big(\frac{kd \log(kd)}{k}\Big) \Big]$$

$$= O(V) \Big[O(\lg^2 V) + O(d \lg^2 V) + O(d \lg d) + O(d \lg kd) \Big]$$

$$= \underbrace{O(Vd)}_{O(E)} \cdot \Big[O(\lg^2 V) + O \cdot (\lg(kd)) \Big]$$

$$\Rightarrow O(E) \cdot \Big[O(\lg^2 V) + O(\lg(kd)) \Big]$$

Things to be Done:

- To show an upper bound on the size of Queue. [It should be sublinear of (V)]
- Find the optimal 'k' deterministically for any given graph.